AI Assisted Coding

Lab Assignment 8.3

Name: K. Charan Yadav

Hall Ticket no: 2303A52367

Batch No: 20

Task -1:

Prompt:

Generate a Python function is_valid_email(email).
Requirements:
- Must contain '@' and '.'
- Must not start or end with special characters
- Should not allow multiple '@'
Also create a version that takes user input and prints result.

```python
# TASK 1: Email Validation
def is_valid_email(email):
    if not isinstance(email, str):
        return False
    if email.count("@") != 1:
        return False
    if email[0] in ".@#" or email[-1] in ".@#":
        return False
    local, domain = email.split("@")
    if "." not in domain:
        return False
    return True
if __name__ == "__main__":
    user_email = input("Enter your email: ")
    if is_valid_email(user_email):
        print("Valid Email ")
    else:
        print("Invalid Email ")
```

OUTPUT:

```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding> &
ent-8.3.py"
Enter your email: charan@gmail.com
Valid Email
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding> &
ent-8.3.py"
Enter your email: charan@@gmail.com
Invalid Email
```

Justification:

VS Code AI helped generate validation logic. Edge cases like multiple '@' and missing '.' were handled. User input version ensures practical testing. Function correctly validates email formats.

Task 2:

Prompt: Create a Python function assign_grade(score).

Grading:

90–100 → A

80–89 → B

70–79 → C

60–69 → D

Below 60 → F

Handle invalid inputs.

Add user input.

```
# TASK 2: Grade Assignment System
def assign_grade(score):
    if not isinstance(score, (int, float)):
        return "Invalid Input"
    if score < 0 or score > 100:
        return "Invalid Input"
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
if __name__ == "__main__":
    try:
        score = float(input("Enter student score: "))
        print("Grade:", assign_grade(score))
    except:
        print("Invalid Input")
```

Output:

```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-8.3.py"
Enter student score: 80
Grade: B
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-8.3.py"
Enter student score: 45
Grade: F
```

Justification:

AI assisted in implementing boundary conditions.

Invalid values like negative or over 100 are handled.

User input allows real-time grading.

System ensures accurate grade classification.

Task 3:

Prompt:

Create a function is_sentence_palindrome(sentence).
Ignore spaces, punctuation, and case.
Add user input version.

```
# TASK 3: Sentence Palindrome Checker
import re
def is_sentence_palindrome(sentence):
    cleaned = re.sub(r'[^a-zA-Z0-9]', '', sentence).lower()
    return cleaned == cleaned[::-1]
if __name__ == "__main__":
    sentence = input("Enter a sentence: ")
    if is_sentence_palindrome(sentence):
        print("It is a Palindrome ")
    else:
        print("Not a Palindrome ")
```

Output:

```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-8.3.py"
Enter a sentence: A man a plan a canal Panama
It is a Palindrome
```

Justification:

AI suggested text normalization using regex.
Case, punctuation, and spacing are ignored.
Ensures accurate palindrome detection.
User-friendly interactive version implemented.

Task -4:

Prompt:

Create a ShoppingCart class with:
add_item(name, price)
remove_item(name)
total_cost()
Add simple user interaction.

```python
# TASK 4: ShoppingCart Class
class ShoppingCart:
    def __init__(self):
        self.items = {}
    def add_item(self, name, price):
        self.items[name] = price
    def remove_item(self, name):
        if name in self.items:
            del self.items[name]
    def total_cost(self):
        return sum(self.items.values())
if __name__ == "__main__":
    cart = ShoppingCart()
    while True:
        print("\n1. Add Item")
        print("2. Remove Item")
        print("3. Show Total")
        print("4. Exit")
        choice = input("Enter choice: ")
        if choice == "1":
            name = input("Enter item name: ")
            price = float(input("Enter item price: "))
            cart.add_item(name, price)
            print("Item added!")
        elif choice == "2":
            name = input("Enter item name to remove: ")
            cart.remove_item(name)
            print("Item removed!")
        elif choice == "3":
            print("Total Cost:", cart.total_cost())
        elif choice == "4":
            break

```

Output:

```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-8.3.py"

1. Add Item
2. Remove Item
3. Show Total
4. Exit
Enter choice: 1
Enter item name: AI-Assisted coding
Enter item price: 250
Item added!
```

Justification:

VS Code AI helped design class structure.

Methods ensure correct addition, removal, and total calculation.

Interactive menu simulates real e-commerce cart.

Empty cart returns 0 automatically.

Task – 5:

Prompt:

Create convert_date_format(date_str)

Input: YYYY-MM-DD

Output: DD-MM-YYYY

Handle invalid formats.

Add user input version.

```python
# TASK 5: Date Format Conversion
from datetime import datetime
def convert_date_format(date_str):
    try:
        date_obj = datetime.strptime(date_str, "%Y-%m-%d")
        return date_obj.strftime("%d-%m-%Y")
    except:
        return "Invalid Format"
if __name__ == "__main__":
    date_input = input("Enter date (YYYY-MM-DD): ")
    print("Converted Date:", convert_date_format(date_input))
```

Output:

```
PS C:\Users\chara\OneDrive\Desktop\Ai-Assisted Coding>
ent-8.3.py"
Enter date (YYYY-MM-DD): 2026-02-11
Converted Date: 11-02-2026
```

Justification:

AI assisted in using datetime for format validation.
Invalid inputs are handled safely using try-except.
Ensures correct date transformation.
User-friendly implementation completed.